Crassula Quantum Dot Simulator Guide

Lucas Orona

2017/10/11

**Introduction to Crassula**

This software is designed to simulate electrostatic gate defined quantum dots in two dimensional electron gases(2DEG). The core of the simulation is based on the paper “Modeling the patterned two‐dimensional electron gas: Electrostatics” by John H. Davies, Ivan A. Larkin and E. V. Sukhorukov. In this paper they find the potential at the 2DEG from a rectangular gate on the surface. This simulation takes DXF files that contain the gates and breaks them into a series of rectangles. The formula found by Davies, et al can then be applied to the gates by applying them to each of their constituent rectangles. One key to the performance of the simulation is that the potential at the 2DEG depends on the product of the voltage applied to the gate and a geometric factor. The geometric factor is computationally intensive to compute but only needs to be found once. After these factors are computed, the gates can be rapidly tuned because it only requires multiplying the geometric factor times the new voltage. By contrast, programs like COMSOL will recompute the equivalent of the geometric factors every time a voltage is changed. This front loading of the computation is convenient because it means that the geometric factors can be calculated while the operator is free to eat lunch, take a nap or fab a sample. With reasonable parameters the front loading time should take about 15 minutes to an hour but this depends on the computer used to run the simulation.

Please report any bugs you find to Lucas.Orona@gmail.com.

**Crassula run script**

The function crassula is called from a crassulaRun file. This file contains all the information about the gates, physical and computation parameters. This script is the only file you ever need to edit to run a simulation. An example is included with the functions.

**DXF files**

You will need at least two DXF files to run a simulation. The first file is saved in the structure files.compBox. This is a rectangle that tells the simulator where to compute the voltage. It would be annoying to compute the voltage everywhere the gates are when the active region is quite small.

Additionally, every layer of gates should have its own DXF files because the distance from the 2DEG is determined by which DXF file the gate is saved in. Every gate should be stored in its own layer with the layer name set as the gate name. Gate names must start with letters and must be a single word. Setting the z value in the DXF file for the gate with set the initial voltage on the gate. These DXF file names should be stored as elements of a cell that are stored in files.gateFiles.

You must use the EXPLODE command on your files in autocad immediately before saving them for simulation. The DXF file only recognizes lines, not polygons. The EXPLODE command turns all types of shapes into lines.

There is one additional optional field, saveFile. This should be set as an empty cell, {}, when not in use. Once you have run the simulation once, you can save the geometric factors and gate voltages to a .mat file. This field can then be set to the string that is the name of the file you saved to, including .mat at the end. This will let you skip computing the geometric factors a second time if you want to play with a simulation later and will let you save specific gate voltages if you have a tuning you like. Note that if you change any parameter, as discussed below, you should not use this option because you will just be loading an old calculation, not rerunning the calculation.

**Computation parameters**

The computation parameters are stored in the compParams structure. The first field is vGrid. This is the grid size for the computation at the 2DEG.

The next field is rectGrid. This determines the grid sized used for meshing the gates. It should be a 1 dimensional matrix that contains the mesh size. Note that this can be useful to speed up computations. You can put small features from one layer into a separate DXF from the large features and give them different grid sizing. All of these distances are given in units of the DXF file. Most DXF files use 1 micron as their base unit. Selecting your grid to be .005 then means that you are gridding at the 5 nm scale.

The final field is unitScale. This converts the DXF unit length to meters. For microns it should be 10^-6, for nanometers it should be 10^-9 and for millimeters it should be 10^-3.

**Material and physical parameters**

These parameters are stored in the physParams structure. The fields are pretty self explanatory.

m=effective mass

hbar=Plancks constant

q=electron charge

EF=Fermi level

Epsilon=electrical permittivity

The field z0s contains the distances between the 2DEG and the gate layers. It should be given in DXF units. It is a 1 dimensional matrix with a height for each DXF file.

**Options**

These are options for the simulation. The first field is gpuSwitch. Matlab can run some computations on the GPUs of CUDA enabled NVIDIA graphics cards. This simulation can take advantage of this capability to drastically reduce computation time for the geometric factors. However, not all computers have this. If you do not have a compatible graphics card, set gpuSwitch to 2. If you do have a compatible graphics card, set gpuSwitch to 0. This will run the computation in parallel on the CPUs and again on the GPUs and compare the time. You should do this with a relatively small file to test. The test will then tell you if you should use the GPUs (gpuSwitch=1) or the CPUs (gpuSwitch=2).

The second field is ploRects. This option is extremely time consuming and should only be run the first time that you analyze a set of DXF files. It plots all the rectangular meshing of your gates. This is good because it allows you to check that the meshing is what you want but it does take a while. Once you have checked a file with a certain mesh size, you should set this to 0 for all future runs.

The final field is indGatePlot. This option lets you plot the potential from each individual gate. This can be useful if you want to see which gate has the most effect on a certain region that multiple gates influence. To turn this on, set it to 1. To turn it off, set it to 0.

**Running a simulation**

Once you have all the above parameters set to the values that you like, run the crassulaRun script. It may take a while for it to compute the geometric factors. I recommend running it with the saveFile option just to ensure that it downloaded correctly the first time. The second time, I recommend running it with the saveFile={} but with the names still as the example names. This should produce the same geometric factors as before.

**Tuning the gates**

When the computation of the geometric factors is complete Figure 1000 and Figure 2 will pop up. Figure 2 plots the potential at the 2DEG. Figure 1000 allows you to change the voltages on the gates. The values in the editable boxes are taken from the z heights of the gates in the DXF files. The names of the gates are taken from the layer on which the gate I saved. When you change the boxes to the voltages that you like, click the “Replot with new voltages” button. This will update Figure 2 with the new gate voltages applied. This can be done as many times as you please. When you are completely through tuning the gates click the “Done tuning” button. This will allow you to move on to the next step.

**Saving**

You will be given the option to save the analysis. If you are happy with the gridding, you should always do this the first time that you run the analysis. This will save the geometric factors and the current gate voltages to a .mat file. You can use this file name in the crassulaRun script so that it does not need to compute the geometric factors in the future. As that computation is the most time intensive step, this saves you having to do the most computationally expensive step repeatedly. You can also save different gate tunings under different names and open them up so that the gates are initialized in a way that you like. Note that this does not change the DXF files at all.

**Compute electron ground state**

This will numerically solve Schrodingers Equation for electrons in your quantum dots. It includes the coloumb interactions between electrons. Selecting this option will open up Figure 4 which initially looks just like Figure 2 and the GUI in Figure 1002. You do not want to simulate the electrons everywhere though, just in your dots. You need to select the area surrounding just the dots that you are interested in simulating. This is done with the “Click to get position” button. You need to click on two opposite corners or a rectangle containing the dots in Figure 2. Figure 4 will then update to include just the selected area. You want this to be a small as reasonably possible for computation speed. Remember that electron probability is exponentially suppressed for large potentials so the error this induces is small. You can change the size of the box as many times as you want by reclicking the “Click to get positions” button. Alternatively, you can also enter values into the boxes below this button. The values will round to the closest grid value to what you enter.

The program will automatically detect the number and position of the quantum dots whenever you select the voltages shown in Figure 4. This means that you don’t have to do this manually. However, if you want to set these manually, you can also enter the number of quantum dots that you are simulating. For a double dot this is 2, for a triple it is 3, etc. After entering that number, you need to click the “Select well locations” button. You then need to click the locations of the wells in Figure 4. Note that changing the range of figure 4 does not affect these values. The well locations are used to determine the charge state of the quantum dots. Selecting these manually will turn off the auto-detection.

When you are happy with the parameters, click the “Run simulation” button. The simulation uses the supplied fermi energy to determine the occupation of the dots. It will tell you the final occupation in the main Matlab window. Figure 300+n (where n is the number of electrons) will plot the electron density. When this is done, you have the option to tune the gate voltages and rerun the simulation.

**Charge stability diagrams**

This will allow you to find the charge states of you quantum dots as a function of two swept gates. This is computationally intense (but thoroughly optimized) so be prepared that it can take a bit. If you want to do this, you should try checking the maximum and minimum values that you want to sweep to before finding the charge stability diagram. I recommend never going above 8 electrons in your system. This number will be increased soon. Going to large numbers of electrons makes the computation really long anyway.

The top of Figure 1003 looks exactly like 1002 and functions the exact same. It will actually use the last values you used when finding the ground state above. You don’t need to change those values if you are happy with them. See the above section for using the top half of the controls.

The bottom half sets the gates that you want to sweep, the maximum and minimum gate voltages to ramp between and the voltage step size.

When you are happy with this, click “Run simulation”. It will tell you the charge states that it calculates while it is computing.

When the simulation is complete figure 1004 will pop up. This figure contains three drop down menus and 1 push button. The three drop down menus allow you to map the occupation of you dots onto the red, blue and green color axes. They are normalized so that zero occupancy is black and maximum occupancy is the reddest, bluest and greenest respectively. This allows you to plot the occupancy of up to three dots on the same plot. Note that you can set multiple dots to the same number. This just maps two color axes to one dot. For instance, setting the blue and red dots to 2 would then make the maximum occupancy of 2 to purple. Setting all three to the same number plots in black and white. The numbers correspond to those in Figure 4.

**Why Crassula?**

Because it is my favorite genus of plants.